Weekly Report

July 15, 2018

1 Work

1. AMTG论文已经从14页缩短到12页，删去了一些图片、一些相关工作的论述和一些不重要的讨论。
2. 基本完成降维论文的内容，还剩下一些参数的实验需要跑。
3. Memory GAN的实验中，IS指标可以从4.36（对标论文）稳定提升到4.48左右，最高可以到4.58，但由于不稳定，很难再次复现。目前还是在尝试不同的方法提升性能。
4. 工作时长：工作日每天10个小时，周末共8个小时，共58个小时。

1.1 工作进度

Table 1: 工作进度

项目	进度	截止时间
图布局方法扩展		7.30
降维	还剩下我们的方法在参数上的实验，下周应该就可以完成	
专利	完成撰写，等待律师回复	
AAAI投稿（Memory GAN）	性能可以略微超过对标论文，但是优势还不够明显	9.1

2 Paper Reading

2.1 Conditional Generative Adversarial Networks for Commonsense Machine Comprehension

文章使用GRU的结构，对文本从文字到语句到文档进行建模，然后使用注意力机制，生成最终机器对文档的理解。

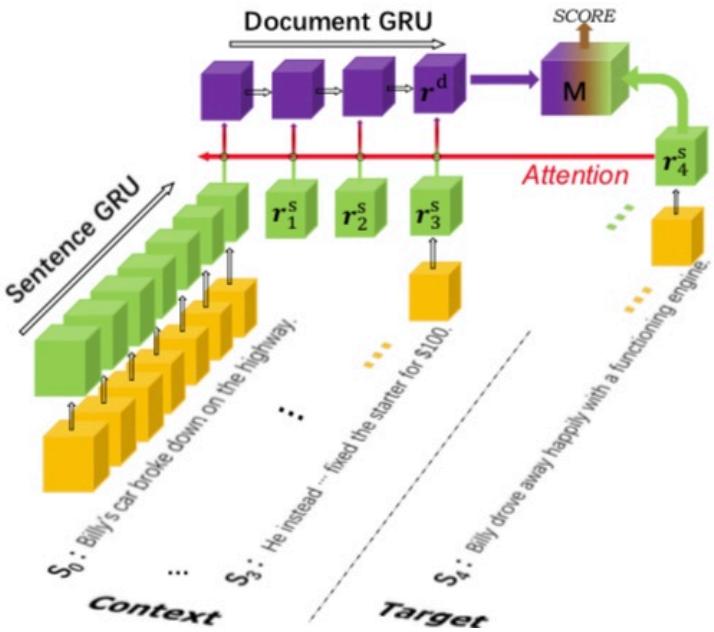


Figure 1: UV-GAN

2.2 Continual Learning with Deep Generative Replay

在多任务的学习中（类似于迁移学习），神经网络会遗忘过去任务中学习到的知识，导致在原始数据集上有性能下降。本文使用GAN的方法，不断生成过去训练时用到的数据，使得网络可以记住过去的知识。

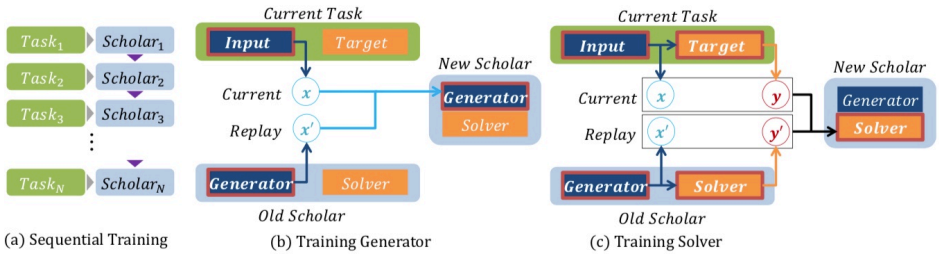


Figure 2: Two Birds with One Stone

2.3 Cross-View Image Synthesis using Conditional GANs

从一张或多张图片生成多角度的其他图片也是目前热点研究的问题（即从有限的二维图片重构三维空间），本文使用GAN的方法，利用两个GAN网络完成了图片生成和语义分割两个任务。

2.4 DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks

在模糊图片上进行object dection的准确率会大幅下降，因此文章使用GAN的方法对原始图片进行去模糊。

Figure 1: DeblurGAN helps object detection. YOLO [30] detections on the blurred image (top), the DeblurGAN restored (middle) and the sharp ground truth image from the GoPro [25] dataset.

Figure 3: SketchyGAN

2.5 GAGAN: Geometry-Aware Generative Adversarial Networks

合成人脸的过程中，人脸的几何属性，比如人脸的landmark，是很重要的一个属性。文章把这个属性作为GAN的一个输入，可以生成不同角度的人脸。

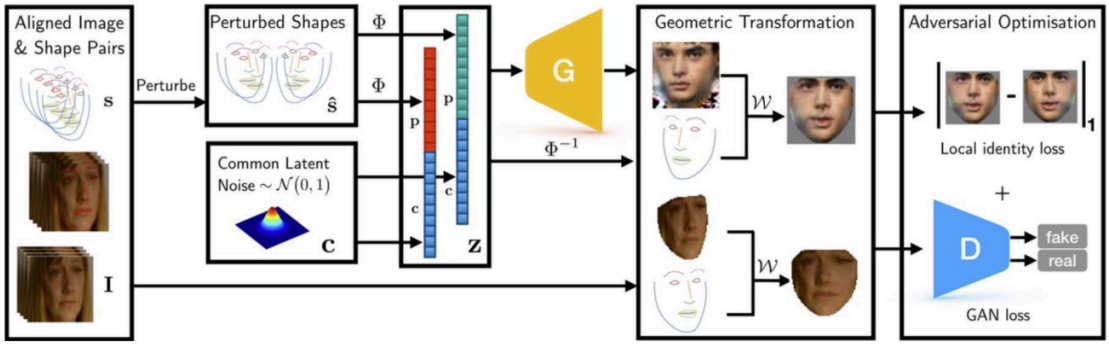


Figure 4: co-attention

Dimensionality Reduction

Michael Shell, *Member, IEEE,* John Doe, *Fellow, OSA,* and Jane Doe, *Life Fellow, IEEE*

Abstract—Visualizing high dimension data is an essential task in visualization and machine learning. Over the past decades, a large number of dimensionality reduction methods have been proposed to generate two-dimensional embedding for data analysis. However, most existing embedding approaches are not scalable to large-scale data. In this work, we introduce an efficient dimensional reduction method with sub-linear computation by incorporating a multi-level representation of the dataset. We construct a *k*-nearest neighbor (KNN) graph from high-dimensional data, generate a series of coarse graphs based on the KNN graph, and iteratively refine the embedding of graphs from coarse to fine scales. Our experiments on various large-scale datasets show the significant acceleration of our method with a comparable embedding quality with LargeVis.

Index Terms—Dimensionality reduction, high-dimensional data, KNN Graph, visualization.

1 Introduction

VISUALIZING high-dimensional data via dimensionality reduction techniques has been widely studied in several fields of science. Dimensionality reduction approaches project high-dimensional data into low-dimension space for visual exploration and inherent structure reveal. In low-dimensional space, adjacent data points tend to share similar attributes and dissimilar data points are far away from each other. Analysts are able to get an overview of data distribution and generate hypotheses before data process. The applications of dimensional reduction include visualization [24], deep learning [16], life science [22], and network analysis [3], [20], etc.

Generally, dimensionality reduction technologies convert the high-dimensional dataset $X = \{x_1, x_2, ..., x_N, x_n \in \mathcal{R}^D\}$ into low-dimensional dataset $Y = \{y_1, y_2, ..., y_N, y_n \in \mathcal{R}^2\}$ for visual analysis. Over the past decades, a large number of dimensionality reduction methods have been proposed. The classical techniques include principal component analysis [15], linear discriminant analysis [6] and multidimensional scaling [19]. To preserve the local properties of the data, some new dimensional reduction methods have been proposed, such as locally linear embedding [26], locality preserving projections [10] and neighborhood preserving embedding [9].

Recently, stochastic neighbor embedding based dimensionality reduction methods have achieved outstanding performance. Maaten and Hinton proposed t-distributed stochastic neighbor embedding (t-SNE) [21] to solve the crowding problem. However, the computational complexity of t-SNE scales quadratically with the size of the dataset. It is time-consuming to visualize large-scale data sets. To accelerate t-SNE, Mattern [31] employed KNN graph and quadtree to speed up the gradient computation with the complexity of $O(NlogN)$. Recently, LargeVis [28] optimizes the objective function with the negative sampling method, which reduces the time complexity in terms of the number of data items $O(N)$. While approximation of the gradient computation using the negative sampling method greatly reduces the computational complexity, it still does not change the fact that LargeVis refines the position of one data point at a time. In addition, we find that LargeVis takes a lot of time to convergence when merging clusters with the same label (see Figure **??**(a)) due to the weak connection between two clusters.

In this paper, we combine a multi-level approach to accelerate convergence with a better initial embedding and share the gradient between similar data points to reduce the gradient computation. First, we accelerate the approximated k-nearest neighborhood (KNN) graph construction by leveraging EFANNA [7]. EFANAA generates an initial KNN graph by KD-tree and refines the graph with the NN-descent. Second, we generate a series of coarse graphs from the KNN graph and iteratively refine the embedding of graphs from coarse to fine scales. Coarse graph represents the global structure of the KNN graph. Each vertex in a coarse graph represents a group of similar data points in the KNN graph. Our approach is able to find a better initial embedding at a low cost based on the coarse graph. The initial embedding of a finer graph is directly derived from the final embedding of a coarser graph. Third, it is time-consuming to compute the gradient of a vertex (a group of similar data points) of a coarse graph by summing over all the gradient of the group. Instead, we approximate the gradient of a coarse graph by sharing the gradient between the group of similar data points. Our approach is performed on various large-scale data sets and achieves significantly acceleration over LargeVis. Experimental results demonstrate that our proposed approach is up to one times faster at KNN graph construction and eight times faster at graph embedding. Our approach achieves a comparable embedding quality compared with LargeVis.

In summary, our contribution includes:

- We propose an efficient dimensionality reduction method by incorporating a multi-level graph embedding approach as well as gradient sharing. The pro-

- *M. Shell was with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332. E-mail: see http://www.michaelshell.org/contact.html*
- *J. Doe and J. Doe are with Anonymous University.*

Manuscript received April 19, 2005; revised August 26, 2015.

posed method scales sub-linearly with the number of input data.
- We conduct experiments on various large-scale data sets. The results show the efficiency of our method with a comparable embedding quality.

The remainder of the paper is organized as follows. In section 2, we review the related embedding approaches. We introduce our method in Section 3. The experimental results are presented in Section 4. Finally, we draw conclusions in Section 5.

2 Related work

As a fundamental means for both visualization and data analysis, dimensionality reduction has been applied in a broad range of fields [27]. The dimensionality reduction approaches are usually classified into two categories: linear and nonlinear methods.

Linear dimensionality reduction methods project high dimension data based on a linear transformation. The distances among data points in original space will be preserved in low dimensional space. Principal component analysis (PCA) [15] is the most popular and widely used method, which minimizes reconstruction error between high-dimensional data and low-dimensional data. The goal of multidimensional scaling (MDS) [19] is preserving pairwise distances between two space. MDS only requires the pairwise distance which is useful in some situations. In addition, it is proofed that MDS is equivalent to PCA in Euclidean space [4]. Likewisethe aim of Sammom mapping [25] is to minimize the distance error which is optimized by steepest descent procedure. When data has associated class labels, linear discriminant analysis (LDA) [6] is employed to reveal label information. LDA maximizes the distance between different clusters and minimizes the distance between data points in each cluster.

Most approaches such as PCA and MDS work well for data on a linear subspace, but they usually fail to detect nonlinear manifold in high-dimensional space. Non-linear distance metric or local structure preservation is employed to capture manifold structure. Isomap [30] estimates the geodesic distance instead of Euclidean distance to minimize the pairwise distance error. Geodesic distance is more suitable to reflect nonlinear low dimension manifold, such as the "swiss roll" dataset. Other nonlinear embedding methods attempt to preserve the local structure: nearby points in high dimensional space remain nearby in the low dimension space. The basic idea of locally linear embedding (LLE) [26] is reconstructing data points with the linear combination of neighbors in high-dimensional space and minimizing the reconstruction error in low-dimensional space. Both Laplacian Eigenmap [1] and locality preserving projections (LPP) [10] attempt to minimize the distance between nearby points. LPP assumes that low dimension embedding is generated by a linear transformation. Therefore, LPP is suitable for new test data. Neural networks such as self-organizing map [18] and autoencoder [12] can also be employed to reduce the dimension.

Unlike previous methods, stochastic neighbor embedding [11] based approaches use probability rather than the distance to measure the similarity among the data points. The goal of these approaches is to minimize the Kullback-Leibler distance between two probability distributions on high-dimensional space and low-dimensional space. t-distributed stochastic neighbor embedding (t-SNE) is proposed to solve the crowing problem [21]. t-SNE shows its significant advantages in generating low-dimensional embedding. However, it is time-consuming to visualize larger data sets with $O(N^2)$ computational complexity. To solve this issue, Maaten proposed Barnes-Hut-SNE (BH-SNE) [31], which employs k-nearest neighborhood graph to estimate the rest probability and approximates distance in low-dimensional space in $O(Nlog(N))$ by the quadtree. Kim et al. introduced an efficient version, called pixel-aligned SNE (PixelSNE) [17]. PixelSNE limits quadtree depth based on screen resolution which guarantees the minimum cell size is not smaller than a pixel. Largevis [28] employs an efficient algorithm to construct the k-nearest neighbor graph and speed up the optimization by negative sampling [23] and edge sampling [29] techniques. LargeVis significantly reduces the computational complexity to linear. Although a number of methods have been developed to speed up t-SNE, there is still much room for further improvement. While approximation of the gradient computation using the negative sampling method greatly reduces the computational complexity, LargeVis still compute the gradient of one data point at a time. We argue that data points within a subgraph can share the gradient. LargeVis takes a lot of time merge clusters with a weak connection. A better initial embedding may overcome local minimums and accelerate the computation of LargeVis. Therefore, we combine a multi-level approach to find a better initial embedding at a low cost.

3 Preliminaries

In this section, we first briefly describe the framework of LargeVis in Section 3.1 and introduce our proposed approach in Section 3.2.

3.1 LargeVis

Though the objective function of LargeVis is defined on a graph, we discuss the optimization from the perspective of minimizing the Kullback-Leibler divergence between the probability distributions of high-dimensional data P and low-dimensional data Q. Due to the relatively high probabilities between adjacent data points, the optimization function will preferentially maintain the distances between neighboring data. The low-dimensional representations are learned by minimizing the Kullback-Leibler divergences between two probability in which the local structure is preserved as much as possible:

$$\begin{aligned} C = KL(P||Q) &= \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \\ &= \sum_i \sum_j p_{ij} \log p_{ij} - p_{ij} \log q_{ij} \end{aligned}$$

Note that the first part of this equation is a constant. Therefore, minimizing the Kullback-Leibler divergence is equal to maximise the following objective function:

$$max \sum_i \sum_j \quad p_{ij} \log q_{ij}$$

The KNN graph is employed to approximate the probabilities between data points in high-dimensional space. Since constructing exact KNN graph is time-consuming, LargeVis proposes an efficient approximate k-nearest neighbor graph construction method. First, an initial k-nearest neighbor graph is constructed based on random projection trees. Then, LargeVis refines the k-nearest neighbor graph by exploring the neighbor's neighbor to improve the accuracy of the KNN graph. Mathematically, the probabilities P are defined as follows:

$$p_{i|j} = \begin{cases} \frac{exp(-d(x_i,x_j)^2/2\sigma_i^2)}{\sum_{k\neq i} exp(-d(x_i,x_k)^2/2\sigma_i^2)} & \text{if } j \in NN_k(x_i) \\ 0 & \text{otherwise} \end{cases}$$

$$p_{ij} = \frac{p_{j|i} + p_{j|i}}{2N}$$

where $d(x_i, x_j)$ is Euclidean distance between high dimension data x_i and x_j, σ_i is the variance of Gaussian distribution on data x_i, $NN_k(x_i)$ denotes the k-nearest neighbors of data x_i, and N is the data size. The parameter σ_i is computed by fitting the perplexity of $p_{i|\cdot}$ to a given perplexity.

A heavy-tailed distribution is employed to measure the probabilities Q in low-dimensional space:

$$q_{ij} = \frac{(1 + d(y_i, y_j)^2)^{-1}}{\sum_{k\neq l}(1 + d(y_i, y_j)^2)^{-1}}$$

Inspired by the negative sampling techniques, to compute the gradient of y_i, LargeVis randomly selects one connected positive sample (y_j) in the KNN graph and M negative samples ($y_{j_k}, k = 1, 2, ..., M$). Thus, the task is to separate the positive point and the negative points. The objective function is reformulated as follows:

$$\max \quad \sum_{i,j} p_{ij}[\log q_{ij} + \sum_{k=1}^{M} \gamma \log(1 - q_{ij_k})]$$

where γ is the parameter for balancing positive and negative weights. The gradient is given by:

$$dy_i = -\frac{2p_{ij}(y_i - y_j)}{1 + d(y_i, y_j)^2} + \sum_{k=1}^{M} \gamma \frac{2p_{ij}(y_i - y_{j_k})}{d(y_i, y_{j_k})^2(1 + d(y_i, y_{j_k})^2)}$$

LargeVis optimizes the objective function with edge sampling method [29] by stochastic gradient descent. Edge sampling method randomly samples two data points (y_i and y_j) based on the probability p_{ij} to avoid excessive gradient.

3.2 Our Method

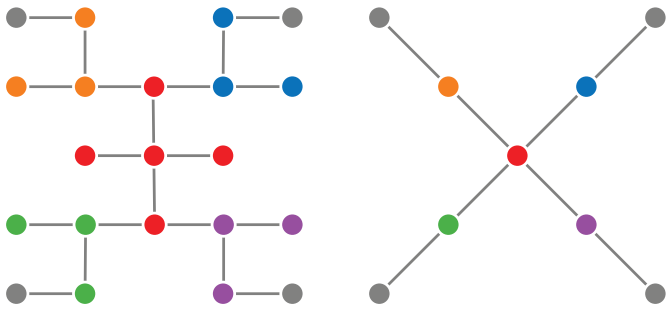
In this section, we seek to present an accelerated algorithm of LargeVis. We employ a faster KNN graph construction technique from EFANNA [7] to accelerate the graph construction and a multi-level approach to generate a better initial embedding for the graph projection. In addition, we share the gradient to speed up the optimization.

KNN Graph Construction We employ the KNN graph construction part of EFANNA [7] to further accelerate the graph construction. The basic idea behind this approach is "a neighbor of a neighbor is also likely to be a neighbor". EFANAA generates an initial graph by KD-tree [2] and refines the graph with the NN-descent [5].

Graph Embedding While LargeVis greatly reduces the computational complexity of the KNN graph embedding by using the negative sampling method, there is still much room for further improvement. First, as shown in Figure **??**(a), it is time-consuming to merge groups due to the weak connection between groups. Whenever a data point moves to another group with a small step, it will be attracted back to the group it belongs. This makes merging groups very difficult due to the instability of negative sampling. Second, LargeVis still computes the gradient of one data point at a time and ignores the fact that similar data in the KNN graph can share the same positive and negative samples in gradient computation. For instance, LargeVis always samples new negative samples to compute the gradient of a data point, resulting in frequent memory access and additional computation. Also, it is time-consuming to compute the gradient of a vertex (a group of similar data points) in a coarse graph by summing over all the gradient of the group in $O(nM)$, where n is the size of the group.

We first discuss how to accelerate convergence with a better initialization with a multi-level approach. We generate a series of coarse graphs from the KNN graph. Our method starts from embedding a coarse graph and iteratively refined the embedding of a finer graph. The initial embedding of a finer graph is directly derived from the final embedding of a coarser graph. The multi-level approach provides a better initial embedding at a low cost and overcomes the local minima.

To solve the second issue, the solution is to reduce the number of negative samples to be sampled as well as share the gradient between similar data. Figure 2 shows the original process of LargeVis. For each data point, LargeVis needs to sample one positive sample and M negative samples. Our basic idea is sharing the negative and positive samples between similar data [14]. It is straightforward to sample once and use these samples repeatedly for gradient calculation (see Figure 2). With the same positive point and negative samples, we are able to compute the gradient of a group of similar data in $O(M)$ by sharing the gradient.



(a) 1-NN graph G^0 (b) The coarse graph G^1

Fig. 1. An example of graph coarsening: (a) the 1NN graph G^0; (b) a coarser graph generated by replacing the vertex with the largest degree and its neighbors into a new vertex.

3.2.1 *The Multi-level Approach*

The multi-level approach has been used widely in many graph embedding methods [8], [13]. Our approach includes

two steps: graph coarsening and iterative refinement. We first generate a multi-level representation to capture the global structure of the KNN graph. Then, we iteratively refine the embedding of the KNN graph from coarse to fine scales. It starts from a coarse graph embedding and iteratively optimizes to a refined embedding.

Graph Coarsening. Given a KNN graph $G^0 = (V^0, E^0)$ with a set of vertexes $V^0 = \{v_1^0, v_2^0, ..., v_N^0\}$ and a set of edges $E^0 \subset V^0 \times V^0$. In this case, each vertex represents a low-dimensional data point: $v_i^0 = y_i$. Each edge is a connection between two vertexes. We denote $e = (v_i^0, v_j^0) \in E^0$ if x_j belongs to the k_1-nearest neighbors of x_i: $x_j \in NN_{k_1}(x_i)$. The degree of a vertex is defined as the number of edges connecting to the vertex. The vertex with a larger degree is more likely to capture the global structure of the KNN graph. The goal of graph coarsening is to generate a series of coarse graphs $G^0, G^1, G^2, ..., G^L$ with decreasing sizes, where G^L is the coarsest graph.

Given a graph $G^l = (V^l, E^l)$, we partition the vertex set of G^l into disjoint subsets and generate a coarser graph G^{l+1}. Each subset is collapsed into a new vertex in the graph G^{l+1}. Each vertex is a group of high-dimensional data points. For instance, if v_i^l and v_j^l are assigned into v_k^{l+1} , then $v_k^{l+1} = v_i^l \bigcup v_j^l$. First, we select a vertex v_i^l with the largest degree. Then, v_i^l and its neighbors are assigned into a new vertex of G^{l+1}. We repeat the above process until all vertexes have been assigned. At last, we add edge $(v_i^l, v_j^l) \in E^l$ in the new graph, if v_i^l and v_j^l are assigned into two different vertexes of G^{l+1}. We stop generating coarse graph when $|V^{l+1}| > 0.8|V^l|$.

Iterative Refinement. For the coarsest graph G^L, we project the graph with random initialization. The optimal embedding of the coarsest graph can be found at a low cost. Once the graph embedding of a coarse graph G^l is generated, the initialization of a finer graph G^{l-1} is derived from G^l. The initial position of a vertex v_i^{l-1} in G^{l-1} is set to be the position of vertex v_j^l of G^l, if v_i^{l-1} is assigned to v_j^l in the graph coarsening step. Then, we recursively refine the embedding coarse to fine scales until the finest graph G^0 is done.

We pre-compute the similarity between vertexes of G^0 just once before the optimization. For the layout of G^l, the gradient of a vertex $v^l = \{y_1, .., y_n\}$ is defined as the average gradient of $y_1, .., y_n$:

$$dv^l = \frac{1}{n}\sum_i dy_i$$

where y_i is the low-dimensional embedding of the high-dimensional data x_i and dy_i is the gradient of y_i. However, it is time-consuming to sum over the gradients, because we need to sample new negative and positive samples for each gradient. (see Figure 2).

3.2.2 Gradient Sharing

Given a vertex v^l which consists of a group of high-dimensional data points $\{y_1, y_2, ..., y_n\}$, we make the following assumptions to share the gradient. We first assume that the data in the group are similar to each other according to the graph coarsening process, which means they can share positive and negative samples and they have a similar probability to other data:

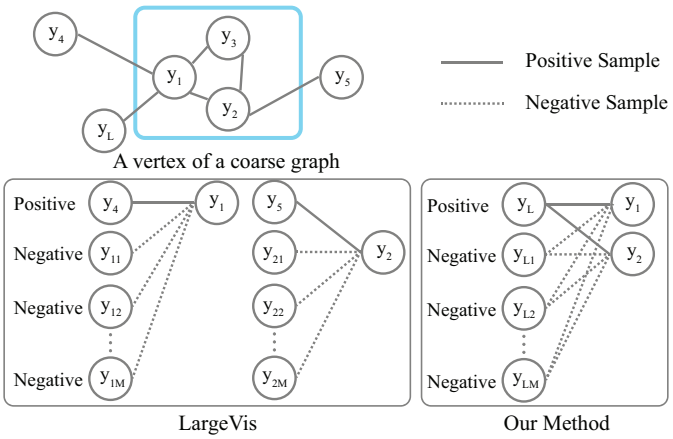


Fig. 2. The KNN graph is constructed on high dimensional space. The solid line represents edge which connects to the positive sample and the dashed line encodes the edge which connects to the negative sample. LargeVis samples new positive and negative samples for each vertex. Our method shares the negative and positive sample in gradient computation of a vertex in a coarse graph.

$$\text{A1}: \quad \|p_{i,k} - p_{j,k}\|_2 \leq \lambda, \forall y_i, y_j \in v^l$$

Next assumption is that the data in the group are close to each other. For the sake of simplicity, we assume that they have the same coordinates.

$$\text{A2}: \quad y_i = y_j, \forall y_i, y_j \in G$$

First, we share the positive y_L and negative points $y_{L_k}, k = 1, 2, ..., M$ in the gradient (A1). Second, according to the assumption (A2), we can approximate the distance between the group to positive samples and negative samples: $d(y_i, y_L) = d(y_j, y_L), d(y_i, y_{L_k}) = d(y_j, y_{L_k})$. The search direction is computed as:

$$\begin{aligned} dy_i &= -\frac{2p_{iL}(y_i - y_L)}{1 + d(y_i, y_L)^2} + \sum_{k=1}^{M} \gamma \frac{2p_{iL}(y_i - y_{L_k})}{d(y_i, y_{L_k})^2(1 + d(y_i, y_{L_k})^2)} \\ &\approx -\frac{2p_{jL}(y_j - y_L)}{1 + d(y_j, y_L)^2} + \sum_{k=1}^{M} \gamma \frac{2p_{jL}(y_j - y_{L_k})}{d(y_j, y_{L_k})^2(1 + d(y_j, y_{L_k})^2)} \\ &= dy_j, \ \forall y_i, y_j \in G \end{aligned}$$

Then, we can share the gradient within the whole group. Therefore, the gradient of the vertex v_i^l can be regarded as the gradient of a random data in the group:

$$dy_i^l = \frac{1}{in}\sum_j dy_{ij} = dy_{iL}$$

Computing the gradient once and assigning the gradient to other data points in the same group will accelerate the optimization process significantly. In practice, we randomly select a data point, compute the gradient and apply the gradient of this data point to the entire group. For instance, given two groups (e.g., G_1 and G_2) with the same labels, we need to move at least $min(|G_1|, |G_2|)$ steps to merge two clusters by moving single data points one by one. We can merge two groups within several steps at the group level using our method.

3.2.3 Computational Complexity

The computational complexity of graph embedding includes coarse graph generation and iterative refinement.

For coarse graph generation, degree computation requires to access all edges of the KNN graph, yielding a computational complexity of $O(kN)$. The worst case of creating G^l from G^{l-1} is accessing all nodes and links, yielding a computational complexity of $O(|V^l| + |E^l|)$. Let us assume that $|V^l| \leq 0.8|V^{l-1}|$ and $|E^l| \leq 0.8|E^{l-1}|$ for all $l = 1, ..., L$, then computational complexity of graph coarsening is $(|E^0| + |V^0|)(1 + \frac{4}{5} + ... + (\frac{4}{5})^L) \leq 5(|E^0| + |V^0|) = 5(k+1)N$. The computational complexity of creating all coarse graphs is linear in $O(kN)$.

For iterative refinement, we randomly select a node and compute the gradient of the node using the negative sampling technique in each iteration. The gradient computation consists of a positive sample and M negative samples. Therefore, each iteration takes $O(\mathrm{M})$ time, where M is the number of negative samples. The number of iterations used for optimization is usually proportional to the number of vertexes. In practice, the iteration number of embedding G^l is $500|V^l|$. The total computational complexity of iterative refinement is $500|V^0|(1+\frac{4}{5}+...+(\frac{4}{5})^L) = O(2500|V^0|\mathrm{M}) = O(2500\mathrm{NM})$. Therefore, the computational complexity of graph embedding is $O(kN + 2500NM)$

4 Experiments

In this section, we show the quantitative and qualitative embedding result of our method through three experiments. We conducted all experiments on a single desktop PC with Intel Xeon E5 1660 CPU, 32GB memory and Ubuntu 17.04 installed.

Datasets The statistics for datasets are summarized in Table 1.

TABLE 1
Summary of data sets

Data set	Size	Dimension	Class number
MNIST	70,000	784	10
Fashion-MNIST	70,000	784	10
CIFAR-10	60,000	1024	10
CIFAR-100	60,000	1024	100
SVHN	630,420	256	10

1) **MNIST:** The MNIST[1] dataset contains 70,000 grey scale handwriting digital images. Each image contains 28 * 28 = 784 pixels and belongs to one of ten digitals from 0 to 9. Each image is considered as a 784-dimension data.
2) **F-MNIST:** The fashion-MNIST[2] is a data set consisting of 70,000 grey scale images with labels from 10 fashion product categories. It shares the same image size with MNIST data set. Fashion-MNIST data set is introduced to replace the original MNIST data set which is too easy for currently machine learning model.
3) **CIFAR-10:** The CIFAR-10[3] data set includes 60,000 color images with 32 * 32 pixels. All images are labeled with 10 classes, such as airplane, bird, etc. There are 6,000 images in each class. We extract features from a powerful convolutional neural network, which consists of seven modules: a 3 × 3 convolution layer followed by batch normalization and a Relu non-linearity. We add max-pool over 2 × 2 patches after every two convolution layers and after the seventh convolution layer. The last layer output with 1024 neurons is fed into a softmax classifier for ten categories. We extract the output of the last layer as a 1024-dimensional feature vector for each image.
4) **CIFAR-100:** The CIFAR-100 data set is similar to CIFAR-10. There are 100 classes with 600 images each. We train a neural network with the same architecture used by CIFAR-10 data set. Each image is represented by a 1024-dimensional vector.
5) **SVHN:** The street view house numbers (SVHN) data set[4] is a real-world house number image data set obtained from Google Street View images. We use all 630,420 color images in character level format where digits are resized to a resolution of 32 * 32 pixels. The SVHN data set is much harder than MNIST to recognize digits. Therefore, We employ a pre-trained deep neural network to preprocess the image and extract better data representation for visualization. The neural network architecture contains seven convolution layers, each of which is a 3 × 3 convolution with batch normalization, ReLU activations and dropout of 0.3. We add max-pool over 2 × 2 patches after every two convolution layers and after the seventh convolution layer. The network then has a final linear layer followed by a softmax for classification. We extract 256-dimensional output of the last convolution layer with as learned representation.

Methods and parameter settings We compare embedding result between our method and LargeVis. After a preliminary evaluation, we employ 2-NN graph to construct the multi-level representation and the total number of iterations to be $1000|V|$ ($10000|V|$ in LargeVis). Both our method and LargeVis, the perplexity to is set as 50, γ is set as 7, and the number of negative samples is $M = 5$. For all dataset, we construct a 100-NN graph as the input of graph embedding. We use the pre-set parameterss of EFANNA for KNN graph construction.

Evaluation Metric We adopt *k*-NN classifier accuracy as the embedding quality metric. We compute the nearest neighborhood classification accuracy based on embedding result. The label of data y_i predicted by kNN classifier is:

$$\overline{l_i} = arg\max_c \sum_{y_j \in N_k(y_i)} I(l_j, c)$$

where l_j is the label of y_j, $N_k(y_i)$ is the set of nearest neighborhoods of y_i and I is identification function. $I(j_j, c)$ = 1 when $l_j = c$, else $I(l_j, c)$ = 0. Therefore, the accuracy is defined as:

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N} I(l_i, \overline{l_i})$$

1. http://yann.lecun.com/exdb/mnist/
2. https://github.com/zalandoresearch/fashion-mnist
3. https://www.cs.toronto.edu/ kriz/cifar.html
4. http://ufldl.stanford.edu/housenumbers/

4.1 Parameter Sensitivity

We report the *10*-NN classifier accuracy with respect to the parameters in our method. Evaluation for other k-NN classifier yeilded similar results, omitted here for space reasons.

Table ? shows the *10*-NN classifier accuracy with respect to the k_1-NN graph used to construct the multi-level representation. We find that the accuracy drops when k becomes large on almost all dataset. The reason is that large nearest neighbors may contains too much noise which results a lower accuracy. For instance, it is more reasonable to merge a vertex and its close neighbors instead of distant neighbors. Therefore, we employ 2-NN graph to capture global structure of KNN graph in our multi-level approach.

TABLE 2
The *10*-NN classifier accuracy with respect to the size of the k_1-NN graph in the multi-level representation.

k_1	1	2	10	50	100
MNIST			0.	0.	0.
F_MNIST			0.	0.	0.
CIFAR-10		0.	0.	0.	0.
CIFAR-100		0.	0.	0.	0.
SVHN		0.	0.	0.	0.

Table ? report the performance with respect to the vertex sampling strategy in the multi-level approach. Generating a new vertex by selecting a vertex with the largest degree (weight selection) results a higher accuracy than random selection. Because the vertex with a larger degree is more likely to capture the global structure of the KNN graph.

TABLE 3
The *10*-NN classifier accuracy with respect to the size of the k_1-NN graph in the multi-level representation.

	Weight Selection	Random Selection
MNIST		
F_MNIST		
CIFAR-10		
CIFAR-100		
SVHN		

Table ? lists the the *10*-NN classifier accuracy with respect to the iteration number. When the iteration number increase, the accuracy converges very fast. The accuracy becomes very stable when the iteration number is large than $1000|V|$.

TABLE 4
The *10*-NN classifier accuracy with respect to the iteration number.

	$100\|V\|$	$500\|V\|$	$1000\|V\|$	$3000\|V\|$	$5000\|V\|$
MNIST			0.	0.	0.
F_MNIST			0.	0.	0.
CIFAR-10		0.	0.	0.	0.
CIFAR-100		0.	0.	0.	0.
SVHN		0.	0.	0.	0.

Table ? reports the *10*-NN classifier accuracy with varied $gamma$. We can see that the accuracy drops when $gamma$ is too large or too small. We employ a trick called "early exaggeration" used in t-SNE to find a better embedding for coarse graph. In pactice, we set $gamma$ to be 1 for the embedding of coarser graphs and set $gamma$ to be 7 for the finest graph.

TABLE 5
The *10*-NN classifier accuracy with respect to the iteration number.

γ	0.01	0.1	1	5	10
MNIST			0.	0.	0.
F_MNIST			0.	0.	0.
CIFAR-10		0.	0.	0.	0.
CIFAR-100		0.	0.	0.	0.
SVHN		0.	0.	0.	0.

4.2 Running Time Comparison

As shown in Table 6, we report the running time on different date sets. We report the running time of KNN Graph construction (GC), graph embedding (GE) as well as total time (Total).

For kNN graph construction part, our method is about 2 times faster than LargeVis on all dataset. For graph embedding part, since we employ a multi-level approach to reduce the total iteration number, our method achieved a 8 times steady acceleration compared to LargeVis on all data sets. For total time, our method shows the significant acceleration against LargeVis in all cases. For example, on small data set such as MNIST, LargeVis takes about 384.73 seconds totally while our method takes only 78 seconds. For larger-scale data set such as Twitter, our methods is 4 times faster than LargeVis. In summary, our methods is more efficient than previous method. Table ? shows that graph emebdding takes the most of runnning time of LargeVis. However, KNN graph construction is the performance bottleneck of our method. A future direction is to design a more efficient KNN graph construction algorithm.

4.3 Embedding Quality

Table 7 compares the classification accuracy by kNN classifier with different k. The KNN classification accuracy gap between LargeVis and ours method is very small, indicating that our method achieves a comparable embedding quality. For F-MNIST and CIFAR-100 data sets, the accuracy of our method is a litter lower. We conjecture the reason is that the multi-level approach cannot capture very accurate global structure of KNN graph for these difficult data. Therefore, the error of sharing the gradient is larger than other data sets, resulting lower accuracy.

4.4 Visualization Comparison

We present the visualization result of LargeVis and our method on various data sets in Figure **??** and Figure **??**. Figure **??** shows the graph embedding process of LargeVis and our method on MNIST data set. For LargeVis, we visualize the graph embedding result in two-dimensional space after every 5% of total iteration number. We also visualize the final embedding result of every coarse graph of our method. Groups are generated at a very early stage and LargeVis spent a lot of time merging groups of the same category (see Figure **??**(a)). In Figure **??**(b), our method can

TABLE 6
Comparison of running times between LargeVis and ours. We show the running time of KNN graph construction (GC), graph embedding (GE) as well as total time (Total) of two algorithms on six data sets. The numbers in parentheses represent the standard deviation of ten repetitions.

Data set	MNIST			F-MNIST			CIFAR-10		
Stage	Graph	Optimization	Total	Graph	Optimization	Total	Graph	Optimization	Total
LargeVis	80.49 (0.53)	295.36 (0.73)	384.73 (0.89)	61.43 (0.42)	296.52 (0.79)	365.92 (0.90)	100.58 (0.58)	278.63 (0.73)	394.07 (0.68)
Ours	39.24 (0.89)	34.37 (0.30)	78.00 (0.85)	35.85 (0.62)	36.61 (0.22)	75.75 (0.56)	41.95 (0.36)	31.85 (0.20)	84.82 (0.51)
Speed up	2.1X	8.6X	4.9X	1.7X	8.1X	4.8	2.4X	8.7X	4.6X
Data set	CIFAR-100			SVHN			Twitter		
Stage	Graph	Optimization	Total	Graph	Optimization	Total	Graph	Optimization	Total
LargeVis	135.03 (0.49)	278.48 (0.68)	428.51 (0.52)	720.06 (5.25)	1306.20 (5.27)	2086.34 (7.91)			
Ours	59.43 (0.76)	32.96 (0.10)	104.02 (0.72)	336.31 (6.49)	176.85 (2.94)	530.06 (7.84)			
Speed up	2.3X	8.4X	4.1X	2.1X	7.4X	3.9X			

TABLE 7
Embedding quality between LargeVis and our method. We report the accuracy of kNN classifier on 2D embedding result. The numbers in parentheses represent the standard deviation of ten repetitions.

Data set	MNIST					F-MNIST					CIFAR10				
k	1	5	10	30	50	1	5	10	30	50	1	5	10	30	50
LargeVis	94.42 (0.26)	96.70 (0.24)	96.72 (0.24)	96.59 (0.28)	96.53 (0.29)	74.59 (0.52)	79.38 (0.51)	80.32 (0.59)	80.85 (0.58)	80.83 (0.57)	94.77 (0.24)	96.54 (0.21)	96.66 (0.22)	96.66 (0.19)	96.64 (0.20)
Ours	94.26 (0.24)	96.62 (0.17)	96.62 (0.15)	96.57 (0.17)	96.55 (0.17)	73.63 (0.71)	78.34 (0.52)	79.35 (0.45)	80.09 (0.55)	80.05 (0.38)	94.58 (0.36)	96.41 (0.23)	96.49 (0.22)	96.50 (0.21)	96.48 (0.24)
	CIFAR100					SVHN					Twitter				
	1	5	10	30	50	1	5	10	30	50	1	5	10	30	50
LargeVis	53.11 (0.62)	59.66 (0.59)	61.47 (0.73)	62.08 (0.64)	61.76 (0.59)	95.46 (0.05)	96.94 (0.06)	97.08 (0.06)	97.14 (0.05)	97.15 (0.05)					
Ours	51.17 (0.46)	57.77 (0.52)	59.64 (0.54)	60.36 (0.71)	60.20 (0.68)	95.36 (0.12)	96.83 (0.08)	96.96 (0.08)	97.03 (0.08)	97.03 (0.08)					

find a better embedding earlier and achieves a comparable embedding result.

Figure **??** shows the visualization of graph embedding result of LargeVis and our method on five datasets. The color of each dot represents the corresponding label. For datasets with clearly category information, the visualization generate by our methods are meaningful. For instance, Figure **??** shows the two-dimensional embedding result on MNIST data set. Ten categories of data points are well separated. In summary, the visualizations are comparable to each other for two methods.

5 Conclusion

In this paper, we present an accelerated version of LargeVis by leveraging space partition method. We explore the idea of optimizing based on the local clusters instead of single data points, which significantly speed up LargeVis. We first employ an efficient version of approximate nearest neighbor method to computer the input similarity on high-dimensional space. Then we accelerate the optimization by sharing the gradient between nearby points in low dimensional space. The experiments on a variety of data sets show the efficiency of our method in comparison with LargeVis. Our method is available at https://gitlab.com/Twilightsnow/DimensionReduction.

In the future, we will develop a GPU version of our method to improve the calculation speed. We plan to apply this idea to other embedding methods, such as t-SNE and word2vec. We will focus on accelerating the graph layout using our method.

Appendix A
Proof of the First Zonklar Equation

Appendix one text goes here.

Appendix B

Appendix two text goes here.

Acknowledgments

The authors would like to thank...
This work is supported by ...

References

[1] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.
[2] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.
[3] HongYun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques and applications. *CoRR*, abs/1709.07604, 2017.
[4] Trevor F Cox and Michael AA Cox. *Multidimensional scaling*. CRC press, 2000.
[5] Wei Dong, Charikar Moses, and Kai Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World wide web*, pages 577–586. ACM, 2011.

[6] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of human genetics*, 7(2):179–188, 1936.
[7] Cong Fu and Deng Cai. Efanna: An extremely fast approximate nearest neighbor search algorithm based on knn graph. *arXiv preprint arXiv:1609.07228*, 2016.
[8] Pawel Gajer and Stephen G Kobourov. GRIP: Graph Drawing with Intelligent Placement. In *International Symposium on Graph Drawing*, pages 222–228. Springer, 2000.
[9] Xiaofei He, Deng Cai, Shuicheng Yan, and Hong-Jiang Zhang. Neighborhood preserving embedding. In *Tenth IEEE International Conference on Computer Vision*, volume 2, pages 1208–1213 Vol. 2, Oct 2005.
[10] Xiaofei He and Partha Niyogi. Locality preserving projections. In *Advances in neural information processing systems*, pages 153–160, 2004.
[11] Geoffrey E Hinton and Sam T Roweis. Stochastic neighbor embedding. In *Advances in neural information processing systems*, pages 857–864, 2003.
[12] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.
[13] Yifan Hu. Efficient, High-Quality Force-Directed Graph Drawing. *Mathematica Journal*, 10(1):37–71, 2005.
[14] Shihao Ji, Nadathur Satish, Sheng Li, and Pradeep Dubey. Parallelizing word2vec in shared and distributed memory. *CoRR*, abs/1604.04661, 2016.
[15] I. T. Jolliffe. *Principal Component Analysis and Factor Analysis*, pages 115–128. Springer New York, New York, NY, 1986.
[16] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.
[17] Minjeong Kim, Minsuk Choi, Sunwoong Lee, Jian Tang, Haesun Park, and Jaegul Choo. Pixelsne: Visualizing fast with just enough precision via pixel-aligned stochastic neighbor embedding. *arXiv preprint arXiv:1611.02568*, 2016.
[18] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, Sep 1990.
[19] Joseph B Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.
[20] O. H. Kwon, T. Crnovrsanin, and K. L. Ma. What would a graph look like in this layout? a machine learning approach to large graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):478–488, Jan 2018.
[21] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.
[22] Ahmed Mahfouz, Martijn van de Giessen, Laurens van der Maaten, Sjoerd Huisman, Marcel Reinders, Michael J. Hawrylycz, and Boudewijn P.F. Lelieveldt. Visualizing the spatial gene expression organization in the brain through non-linear similarity embeddings. *Methods*, 73(Supplement C):79 – 89, 2015. Spatial mapping of multi-modal data in neuroscience.
[23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
[24] N. Pezzotti, B. P. F. Lelieveldt, L. v. d. Maaten, T. Hllt, E. Eisemann, and A. Vilanova. Approximated and user steerable tsne for progressive visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 23(7):1739–1752, July 2017.
[25] J. W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18(5):401–409, May 1969.
[26] Lawrence K Saul and Sam T Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of machine learning research*, 4(Jun):119–155, 2003.
[27] Carlos Oscar Sánchez Sorzano, Javier Vargas, and A Pascual Montano. A survey of dimensionality reduction techniques. *arXiv preprint arXiv:1403.2877*, 2014.
[28] Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th International Conference on World Wide Web*, pages 287–297. International World Wide Web Conferences Steering Committee, 2016.
[29] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.
[30] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.
[31] Laurens Van Der Maaten. Accelerating t-sne using tree-based algorithms. *Journal of machine learning research*, 15(1):3221–3245, 2014.

PLACE PHOTO HERE

Michael Shell Biography text here.

John Doe Biography text here.

Jane Doe Biography text here.